FIG. 1

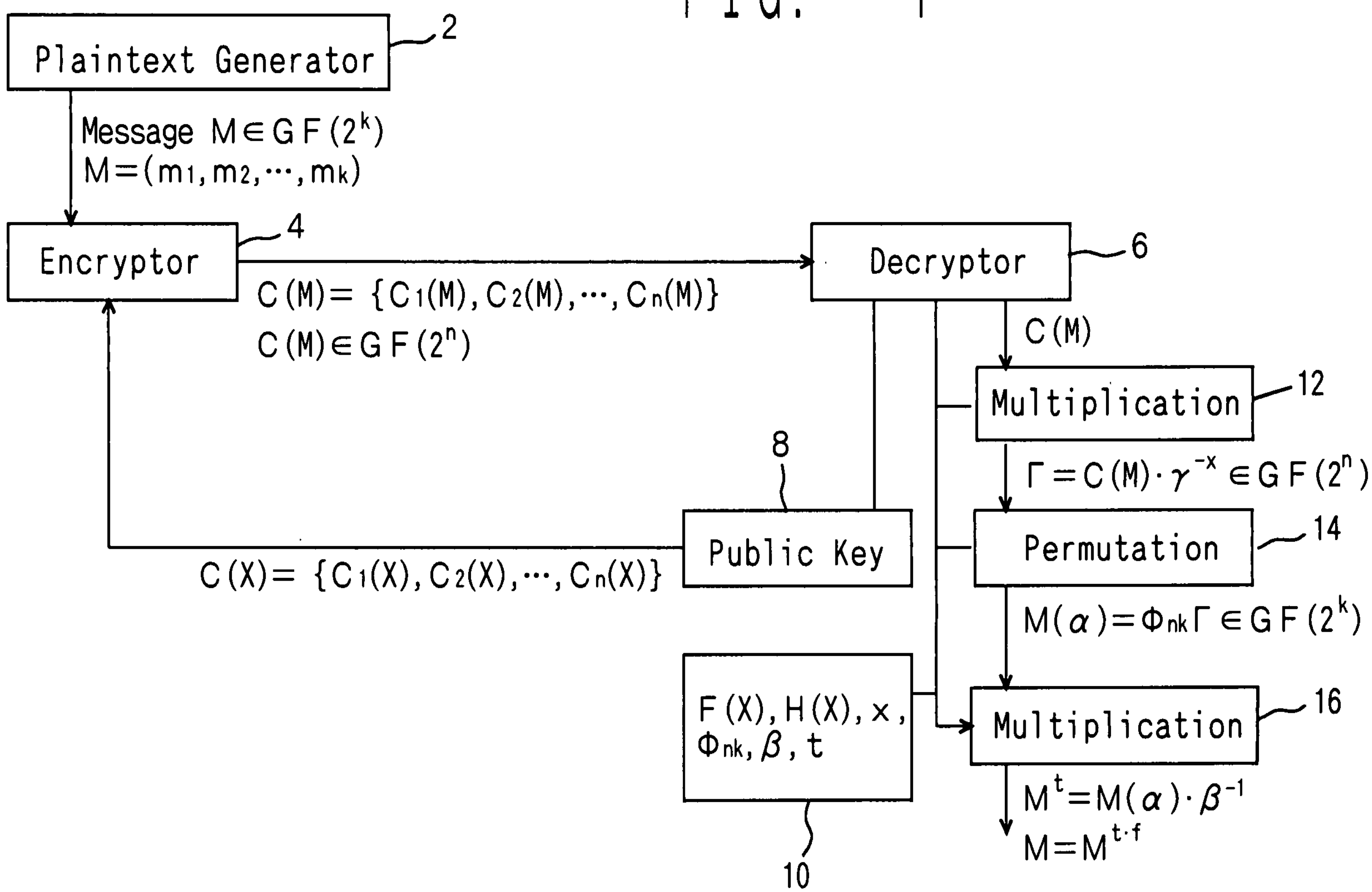

Plaintext Generator
2
Message $M \in GF(2^k)$
$M=(m_1, m_2, \cdots, m_k)$
Encryptor
4
$C(M)= \{C_1(M), C_2(M), \cdots, C_n(M)\}$
$C(M) \in GF(2^n)$
Decryptor
6
$C(M)$
Multiplication
12
$\Gamma = C(M) \cdot \gamma^{-x} \in GF(2^n)$
Permutation
14
$M(\alpha) = \Phi_{nk} \Gamma \in GF(2^k)$
Multiplication
16
$M^t = M(\alpha) \cdot \beta^{-1}$
$M = M^{t \cdot f}$
Public Key
8
$C(X)= \{C_1(X), C_2(X), \cdots, C_n(X)\}$
$F(X), H(X), x,$
Φ_{nk}, β, t
10

FIG. 2

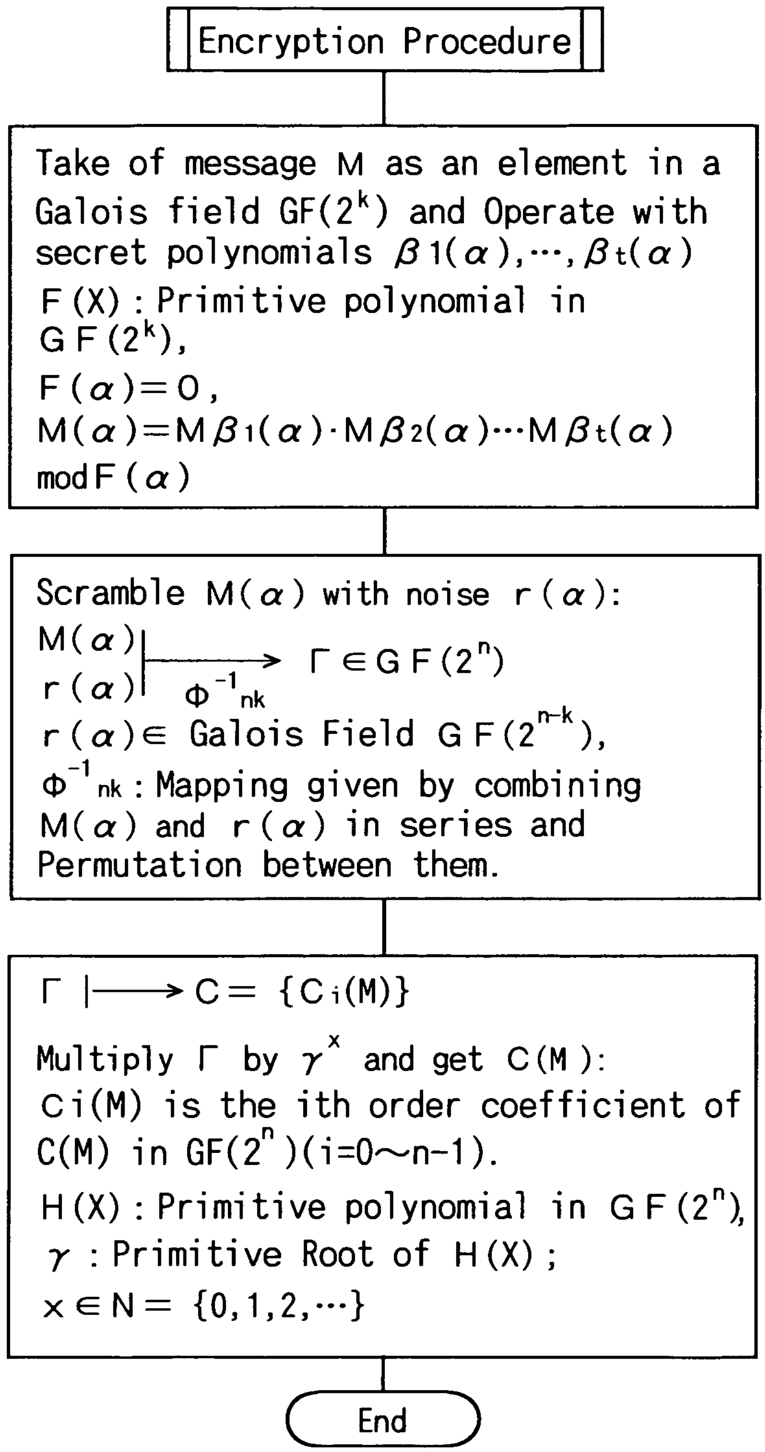

Encryption Procedure
Take of message M as an element in a Galois field GF(2^k) and Operate with secret polynomials β1(α),…,βt(α)
F(X): Primitive polynomial in GF(2^k),
F(α)=0,
M(α)=Mβ1(α)·Mβ2(α)…Mβt(α) mod F(α)
Scramble M(α) with noise r(α):
M(α)
r(α)
Φ^-1 nk
Γ∈GF(2^n)
r(α)∈ Galois Field GF(2^n-k),
Φ^-1 nk: Mapping given by combining M(α) and r(α) in series and Permutation between them.
Γ ⟼ C= {Ci(M)}
Multiply Γ by γ^x and get C(M):
Ci(M) is the ith order coefficient of C(M) in GF(2^n)(i=0~n-1).
H(X): Primitive polynomial in GF(2^n),
γ: Primitive Root of H(X);
x∈N= {0,1,2,…}
End

FIG. 3

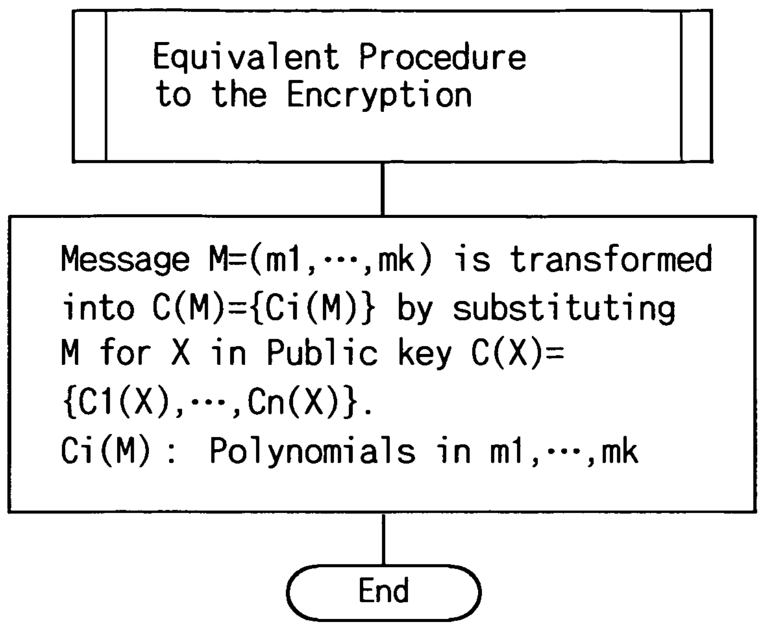

Equivalent Procedure to the Encryption
Message M=(m1,···,mk) is transformed into C(M)={Ci(M)} by substituting M for X in Public key C(X)= {C1(X),···,Cn(X)}.
Ci(M) : Polynomials in m1,···,mk
End

FIG. 4

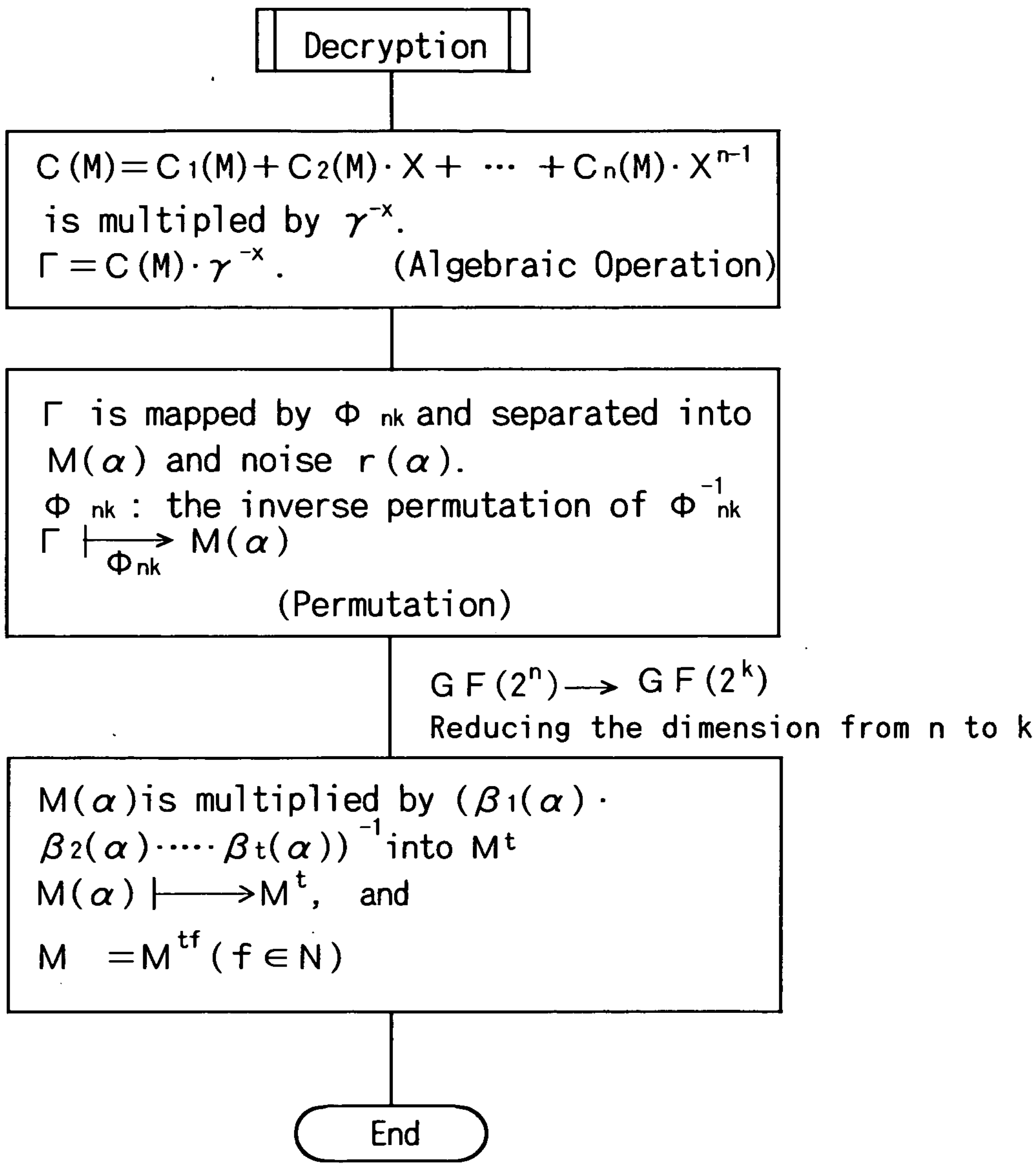

Decryption
C (M)=C1(M)+C2(M)·X+ ··· +Cn(M)·X^n-1
is multipled by γ^-x.
Γ=C (M)·γ^-x.
(Algebraic Operation)
Γ is mapped by Φ nk and separated into
M(α) and noise r(α).
Φ nk : the inverse permutation of Φ^-1 nk
Γ ⟼ M(α)
Φnk
(Permutation)
G F (2^n)⟶ G F (2^k)
Reducing the dimension from n to k
M(α)is multiplied by (β1(α)·
β2(α)·····βt(α))^-1 into M^t
M(α) ⟼M^t, and
M =M^tf (f∈N)
End

FIG. 5

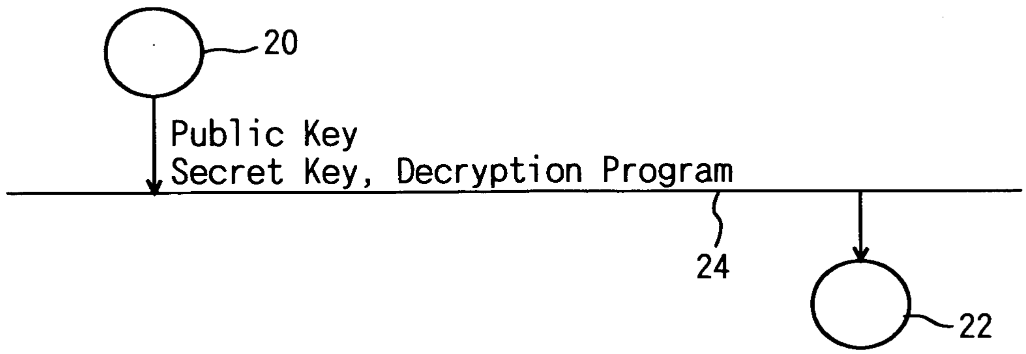
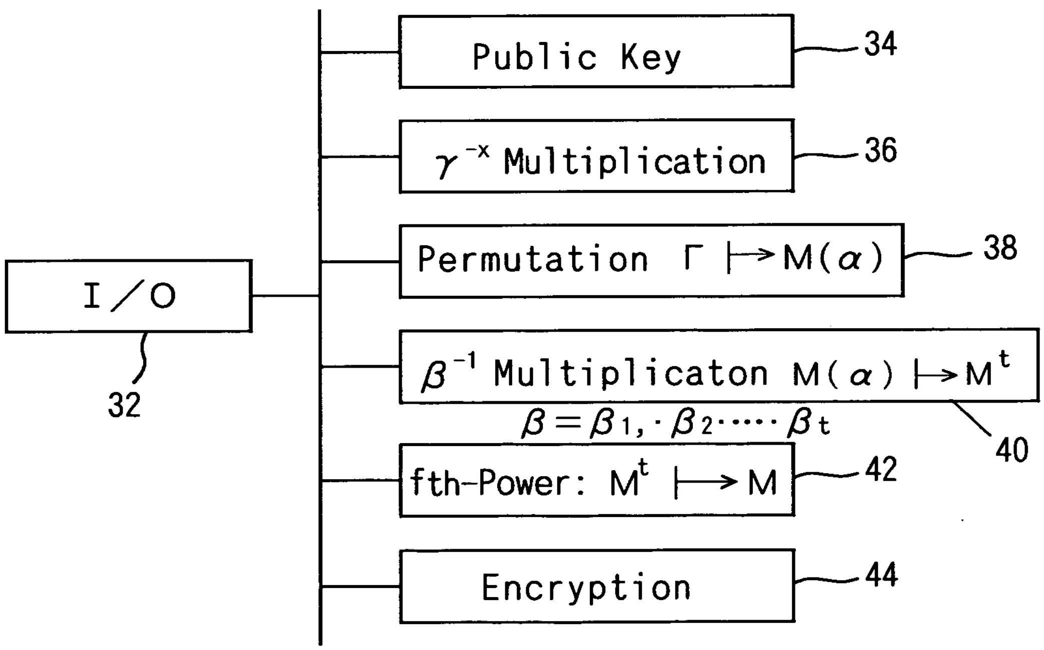

20
Public Key
Secret Key, Decryption Program
24
22

FIG. 6


Public Key
34
γ^-x Multiplication
36
Permutation Γ ⊢→ M(α)
38
I/O
32
β^-1 Multiplicaton M(α) ⊢→ M^t
β = β1, · β2 ····· βt
40
fth-Power: M^t ⊢→ M
42
Encryption
44

30